Department of Computer Science
Franklin College of Arts and Sciences
UNIVERSITY OF GEORGIA

AWAPart: Adaptive Workload-Aware Partitioning of Knowledge Graph

Amitabh Priyadarshi and Krzysztof J. Kochut

Presenter:
Amitabh Priyadarshi
amitabh.priyadarshi@uga.edu
Department of Computer Science
University of Georgia
Athens, Georgia, USA

Resume

Amitabh Priyadarshi
amitabh.priyadarshi@uga.edu
Department of Computer Science
University of Georgia
Athens, Georgia. USA

Highly motivated computer science PhD. Student, skilled in problem solving with a solid understanding of programming practices including full stack development, Machine Learning etc.

- Research Interest: Partitioning the Knowledge Graph to improve performance of various graph operations.
- Skills:
  - Java, Python
  - SQL, SPARQL, Graph database
  - Machine Learning
  - Java Script, jQuery, AJAX etc.
- Experience:
  - Enterprise Information Technology Services (UGA), GA, USA (2016-now)
  - Complex Carbohydrate Research Center (UGA), GA, UGA (2014-2015)
  - Oxient Technology , Pune, India(2009-2013)
- Education:
  - Pursuing PhD. In CS from University of Georgia, Athens, GA,USA
  - Master in Computer Application, Bharti Vidyapeeth University, Pune ,India
  - Bachelor in Computer Application, Indira Gandhi National University, Delhi, India

Department of Computer Science
Franklin College of Arts and Sciences
UNIVERSITY OF GEORGIA

Overview

- Graph Partitioning Problem
- Background
- AWAPart
- Implementation
- Experiment
- Future Work

Graph Partitioning Problem

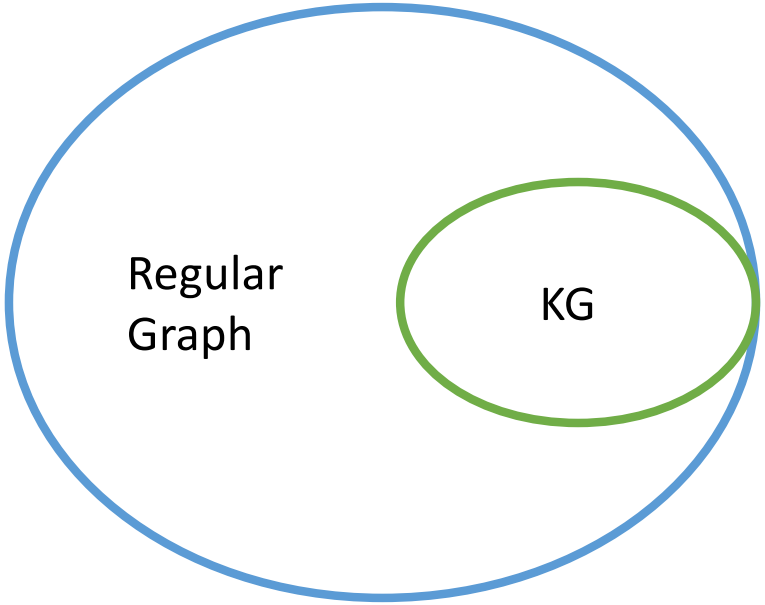
- Graph partitioning is to splitting the graph G into smaller blocks, where connection between blocks are kept at minimum.
- Given
  - Graph G = (V, E) and Number of Partition P
- Output
  - $V = V_0 \cup V_1 \cup V_2 \cup \ldots \cup V_{P-1}$
  - Where:
    - $\{ V_i , V_j\}$ are disjoint $\Rightarrow V_i \cap V_j = \emptyset$
    - $\{ V_i , V_j\}$ are roughly balanced $\Rightarrow |V_i| \sim |V_j|$
    - Minimize $|E_{cut}|$ where $E_{cut} \equiv \{(u, v) \mid u \in V_i , v \in V_j , i \neq j \}$

Background

- Regular Graph vs. Knowledge Graph
- Resource Description Framework
- SPARQL Query Language
- Federated SPARQL Query

Regular Graph vs. Knowledge graph

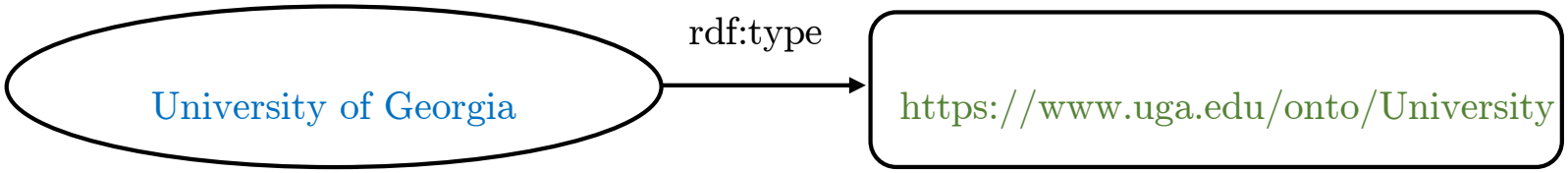
- Graphs includes undirected, directed, weighted, labelled, connected, multigraph etc.
- Knowledge graph is based on a labelled directed graph.



Resource Description Framework (RDF)

- RDF enables embedding of machine-readable information
- RDF statement (triple) has three parts (Subject, Predicate and Object)
  - Subject: resource
  - Predicate : property
  - Object: value

Statement: University of Georgia is a University

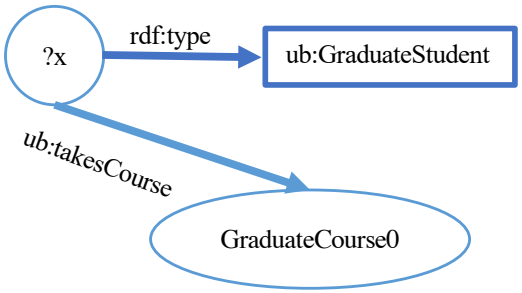


- Popular formats in which RDF data can be serialized
  - Turtle, N-Triples / N-Quads, Notation 3, RDF/XML, RDF/JSON etc.

SPARQL Query Language

- SPARQL used for Querying the data in RDF graphs
- SPARQL tries to match a triple pattern in a RDF graph.
- Triple patterns are like a triple, except that any parts of a triple(s, p, o) can be a variable.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#>
SELECT ?X FROM <LUBM-Synthetic-Data>
WHERE
{
   ?X rdf:type ub:GraduateStudent.
   ?X ub:takesCourse <http://www.Department0.University0.edu/GraduateCourse0>
}
ORDER BY ?X DESC(?X)
LIMIT 10
```

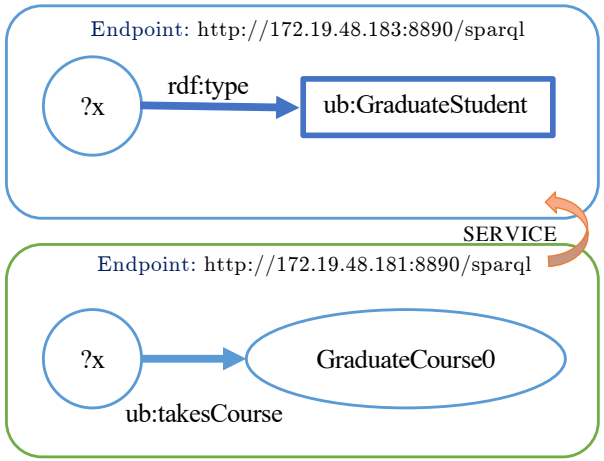


Federated SPARQL Query

- Federated SAPARQL Query
  - It allow to combine solutions from different RDF datasets.
  - SERVICE keyword is used in the WHERE clause that directs a portion of a query towards a particular SPARQL endpoint.

```
SPARQL Endpoint: http://172.19.48.181:8890/sparql  where this query will execute.

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#>
SELECT ?X FROM <LUBM-Synthetic-Data>
WHERE
{
   SERVICE <http://172.19.48.183:8890/sparql>{?X rdf:type ub:GraduateStudent.}
   ?X ub:takesCourse <http://www.Department0.University0.edu/GraduateCourse0>
}
ORDER BY ?X DESC(?X)
LIMIT 10
```



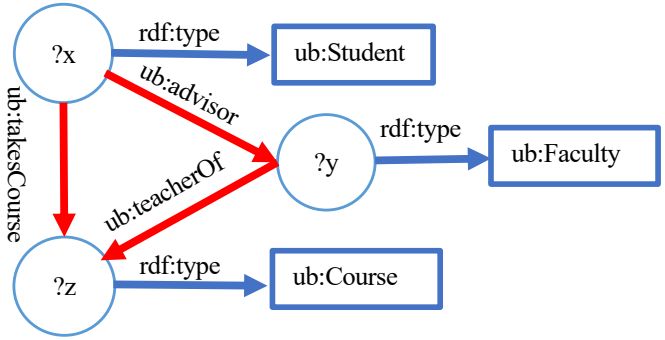
Department of Computer Science
Franklin College of Arts and Sciences
UNIVERSITY OF GEORGIA

Adaptive Workload-Aware Partitioning of Knowledge Graph

- Primary Goal:
  - Adaptive Graph partitioning system that partition based on changing workloads.
- Secondary Goal:
  - Handle Very Large Graphs
  - No Replication
  - Performance Gain
  - Adaptive Workload Aware System

Graph Indexing, Features Extraction and Analyzing Workload

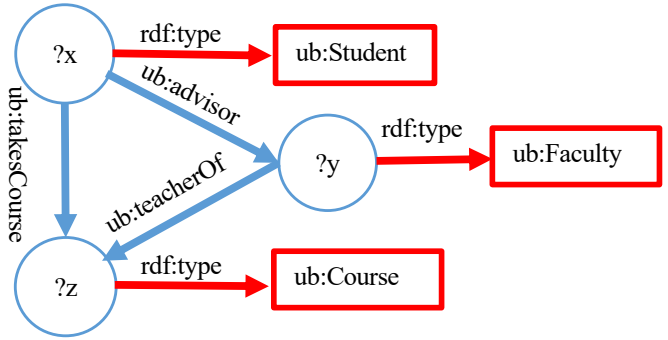
- Indexing the datasets(Knowledge Graph) is used to search for specific sets of pattern.
- Features category can be identified in graph pattern.
  - Predicate



Graph Indexing, Features Extraction and Analyzing Workload

- Indexing the datasets(Knowledge Graph) is used to search for specific sets of pattern.
- Features category can be identified in graph pattern.
  - Predicate
  - Predicate-Object

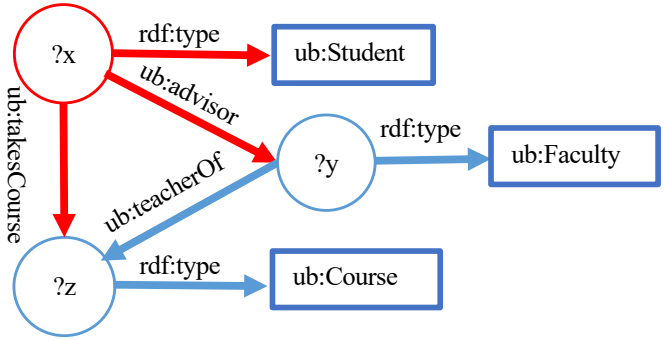
Used for Clustering



Graph Indexing, Features Extraction and Analyzing Workload

- Indexing the datasets(Knowledge Graph) is used to search for specific sets of pattern.
- Features category can be identified in graph pattern.
  - Predicate
  - Predicate-Object
  - Subject-Subject Join

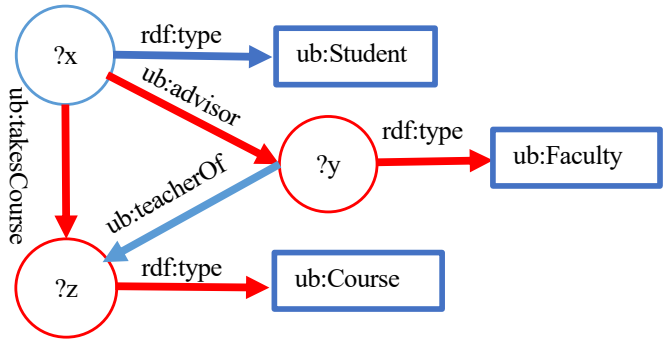
Used for Clustering



Graph Indexing, Features Extraction and Analyzing Workload

- Indexing the datasets(Knowledge Graph) is used to search for specific sets of pattern.
- Features category can be identified in graph pattern.
  - Predicate
  - Predicate-Object
  - Subject-Subject Join
  - Object-Subject Join

Used for Clustering

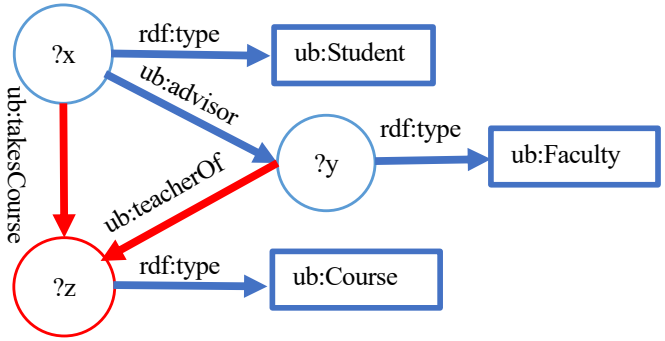


Graph Indexing, Features Extraction and Analyzing Workload

- Indexing the datasets(Knowledge Graph) is used to search for specific sets of pattern.
- Features category can be identified in graph pattern.
  - Predicate
  - Predicate-Object
  - Subject-Subject Join
  - Object-Subject Join
  - Object-Object Join

Used for Clustering

Used for keeping stats about features

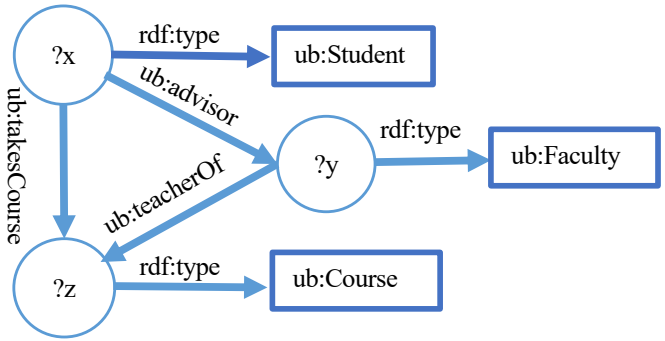


Graph Indexing, Features Extraction and Analyzing Workload

- Indexing the datasets(Knowledge Graph) is used to search for specific sets of pattern.
- Features category can be identified in graph pattern.
  - Predicate
  - Predicate-Object
  - Subject-Subject Join
  - Object-Subject Join
  - Object-Object Join

Used for Clustering

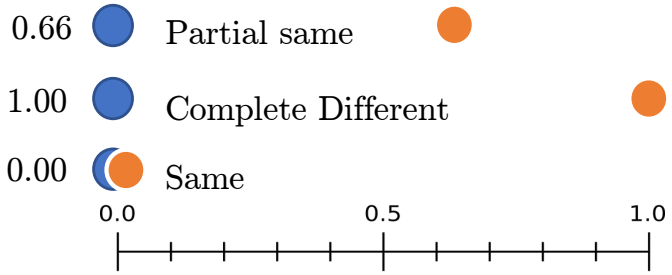
Used for keeping stats about features

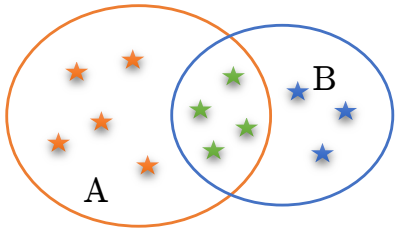


- Query analysis
  - Extraction of features from workload
  - Creation of Metadata of Features and Stats

Distance Matrix and Hierarchical Clustering

- Jaccard Distance matrix: $(1- J_{sim}) = 1- |A \cap B| / |A \cup B|$

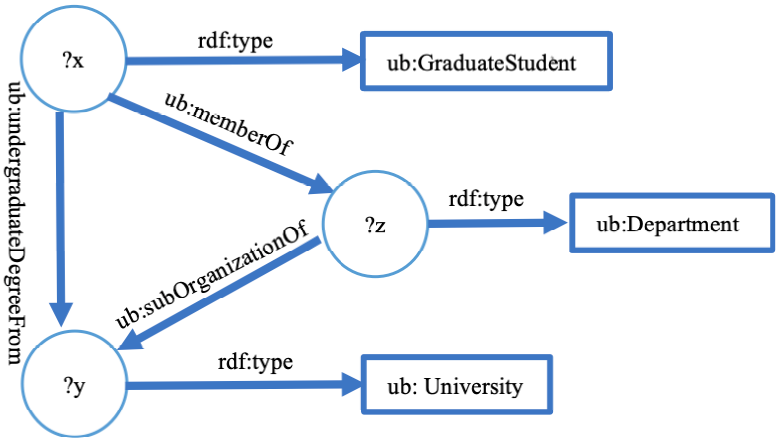




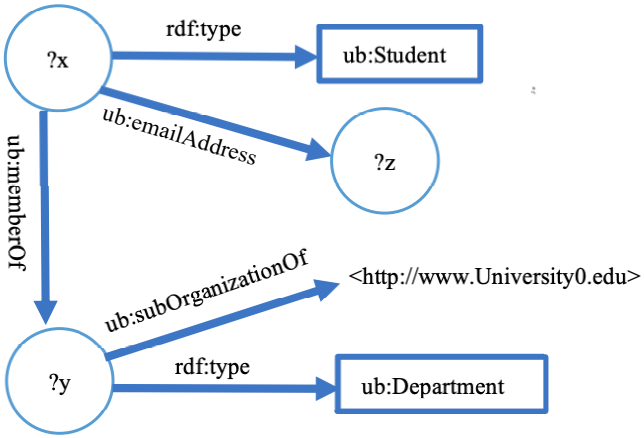
1-J_{SIM} of sets A and B is $1 - (4/12) = 1\text{-}0.33 = 0.67$

- Hierarchical Agglomerative Clustering
  - Cluster grouping : Based on shortest distance among all pairwise distances between queries.
    - The first similarity measure is provided by the initial distance matrix.
    - Once the two most similar queries are grouped together, the distance matrix is recalculated.
    - Recalculation of the distance matrix is based on the choice of the linkage (single, complete, or average).
  - These steps are repeated until there is only one cluster left.

Q2:

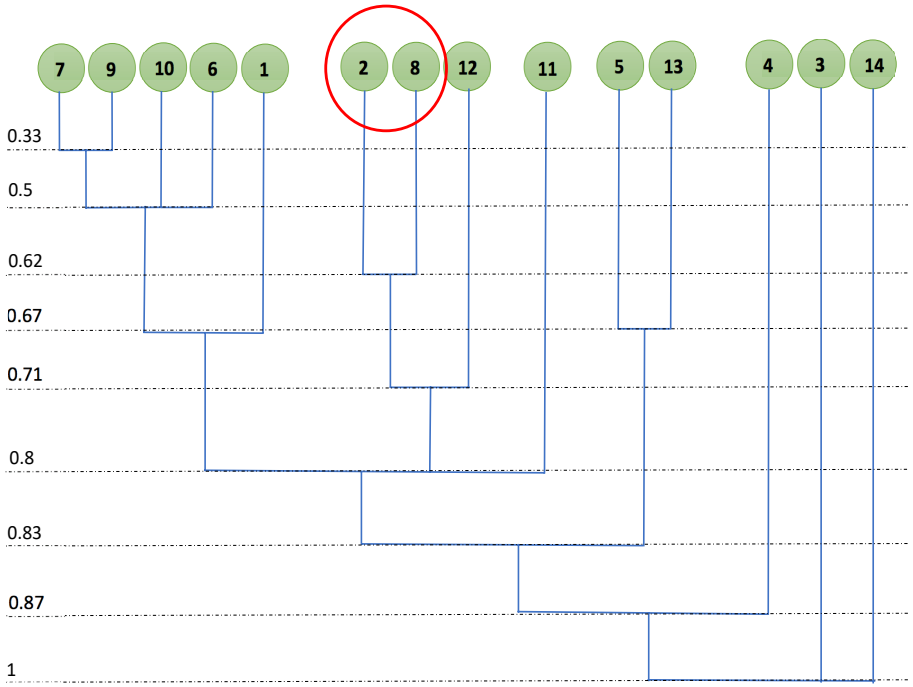


Q8:



- Query 2 has 6 features:
  - 3 *PO* features:
    - *rdf:type → ub:GraduateStudent*
    - *rdf:type → ub:Department*
    - *rdf:type → ub:University*
  - 3 *P* features:
    - *ub:memberOf*
    - *ub:subOrganizationOf*
    - *ub:underGraduateDegreeFrom*
- Query 8 has 5 features:
  - 3 *PO* features:
    - *rdf:type → ub:Student*
    - *rdf:type → ub:Department*
  - 3 *P* features:
    - *ub:memberOf*
    - *ub:subOrganizationOf*
    - *ub:emailAddress*

The distance between these queries is
$(1- J_{sim}) = 1- |Q2 \cap Q8| / |Q2 \cup Q8| = 1- 3/8= 0.625$



Adaptive Partitioning

- Input
  - Initial Knowledge Graph Partition*
  - HAC Dendrogram
  - Features Metadata
- Create Feature set of workload based on distance
- Feature statistics to remove Replication and Balance the Partition
  - **PEER** Features which moves together or Tightly connected features
  - **SIZE** of replicated features and theirs peer features
  - **Dependency** of Queries on these features.
  - Number of distributed **JOINS** it creates on each partition
  - **Frequency** of queries in Workload
- Search for features in graph and create partition based on feature sets.

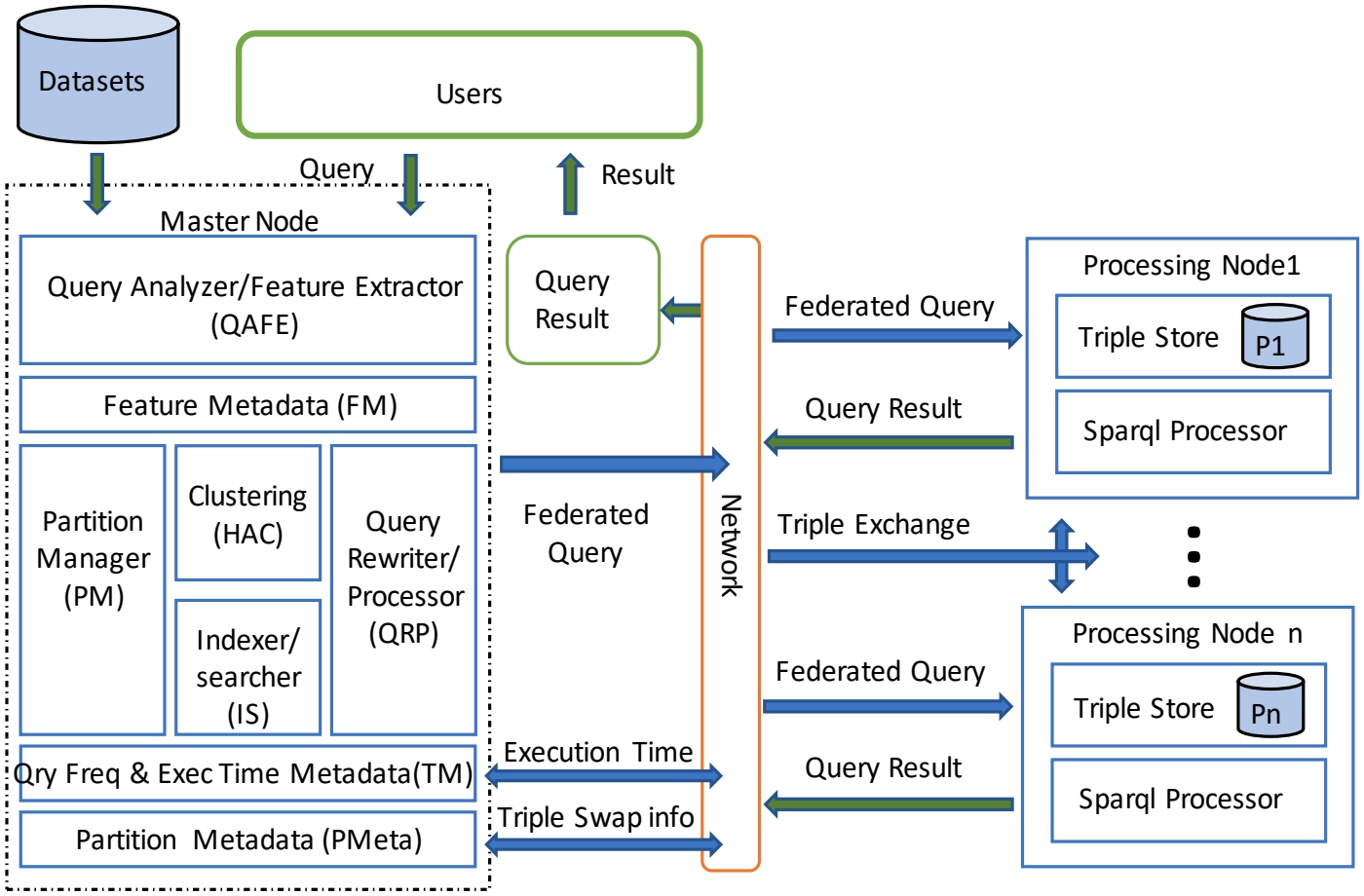
* A. Priyadarshi and K. J. Kochut, "WawPart: Workload-Aware Partitioning of Knowledge Graphs," Cham, 2021: Springer International Publishing, in Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices, pp. 383-395.

Stats

Predicate	ClusterID	ClusterTotalTriple	Cluster triple % allowed versus actual	predicate triple size #count	predicate triple percent / Total	predicate triple percent / Cluster	peer predicate triple count	peer predicate triple count percent / Total	peer predicate triple count percent / Cluster	total predicate in this Cluster	peer predicate count	Unique peer predicate count	peer predicate count percent / Total	Unique peer predicate count percent / Total	peer predicate count percent / Cluster	Unique peer predicate count percent / Cluster	Appear in how many Query	Appear in how many Query in assigned cluster	Frequency of query	Other predicate member in that query
advisor	A	50,578,673	155	4,069,536	2.50	8.05	44,010,540	27.0	87.0	7	5	5	12.5	12.5	71.4	71.4	1	1	1	Student, Faculty, Course, teacherOf, takesCourse
AssistantProfessor	E	11,060,463	34	188,293	0.12	1.70	0	0.0	0.0	16	0	0	0.0	0.0	0.0	0.0	0	0	1	null
AssociateProfessor	E	11,060,463	34	237,740	0.15	2.15	0	0.0	0.0	16	0	0	0.0	0.0	0.0	0.0	0	0	1	null
Chair	D	39,637,983	122	0	0.00	0.00	1	0.0	0.0	10	0	0	0.0	0.0	0.0	0.0	1	1	1	Department, worksFor, subOrganizationOf
Course	A	50,578,673	155	2,152,141	1.32	4.26	45,927,935	28.2	90.8	7	9	5	22.5	12.5	128.6	71.4	2	2	1	Faculty, advisor, teacherOf, Student, takesCourse
degreeFrom	E	11,060,463	34	4,682,313	2.88	42.33	0	0.0	0.0	16	0	0	0.0	0.0	0.0	0.0	0	0	1	null
Department	**B**	**39,637,983**	**122**	**19,831**	**0.01**	**0.05**	**39,618,152**	**24.3**	**99.9**	**10**	**12**	**9**	**30.0**	**22.5**	**120.0**	**90.0**	**3**	**3**	**1**	**GraduateStudent, emailAddress, University, Chair, works**
doctoralDegreeFrom	E	11,060,463	34	713,785	0.44	6.45	0	0.0	0.0	16	0	0	0.0	0.0	0.0	0.0	0	0	1	null
emailAddress	**B**	**39,637,983**	**122**	**11,061,600**	**6.80**	**27.91**	**22,150,430**	**13.6**	**55.9**	**10**	**4**	**4**	**10.0**	**10.0**	**40.0**	**40.0**	**2**	**1**	**1**	**Student, Department, memberOf, subOrganizationOf**
emailAddress	**D**	**75,632,316**	**232**	**11,061,600**	**6.80**	**14.63**	**33,587,130**	**20.6**	**44.4**	**10**	**4**	**4**	**10.0**	**10.0**	**40.0**	**40.0**	**2**	**1**	**1**	**Professor, worksFor, name, telephone**
Employee	E	11,060,463	34	720,628	0.44	6.52	0	0.0	0.0	16	0	0	0.0	0.0	0.0	0.0	0	0	1	null
Faculty	A	50,578,673	155	720,628	0.44	1.42	47,359,448	29.1	93.6	7	5	5	12.5	12.5	71.4	71.4	1	1	1	Student, Course, advisor, teacherOf, takesCourse
FullProfessor	E	11,060,463	34	168,608	0.10	1.52	0	0.0	0.0	16	0	0	0.0	0.0	0.0	0.0	0	0	1	null
GraduateCourse	E	11,060,463	34	1,071,038	0.66	9.68	0	0.0	0.0	16	0	0	0.0	0.0	0.0	0.0	0	0	1	null
GraduateStudent	A	50,578,673	155	2,498,597	1.54	4.94	28,548,443	17.5	56.4	7	1	1	2.5	2.5	14.3	14.3	2	1	1	takesCourse

Implementation

- Architecture



Department of Computer Science
Franklin College of Arts and Sciences
UNIVERSITY OF GEORGIA

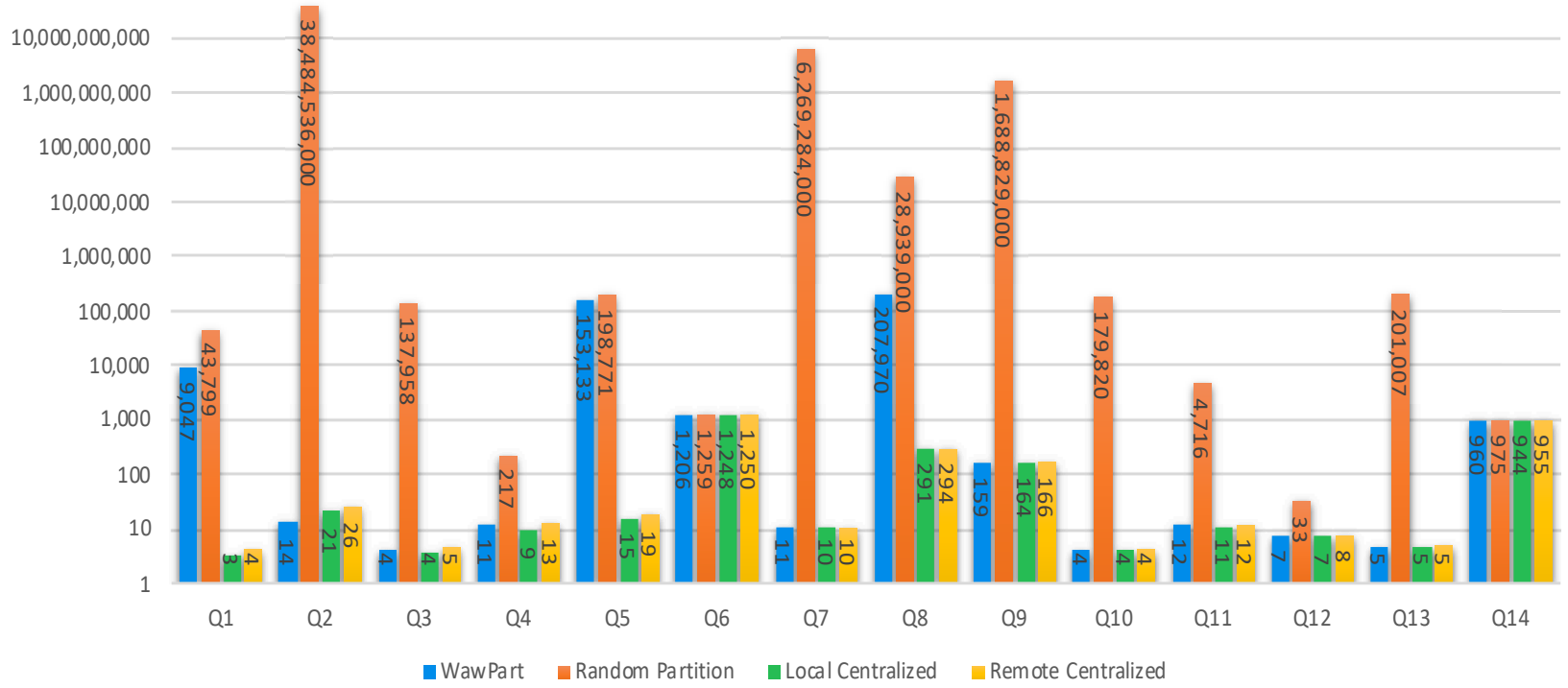
Experiment Setting

- For this experiment we have chosen
  - LUBM dataset of 10 universities with 1,563,927 triples.
  - 14 old LUBM Queries
  - 10 new mixed LUBM Queries
- WawPart as initial partition*.
- A cluster of Intel i5-based systems running Linux Ubuntu 18.04.4 LTS 64-bit OS.
- Openlink Virtuoso RDF triple store.

* A. Priyadarshi and K. J. Kochut, "WawPart: Workload-Aware Partitioning of Knowledge Graphs," Cham, 2021: Springer International Publishing, in Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices, pp. 383-395.

LUBM: Lehigh University Benchmark

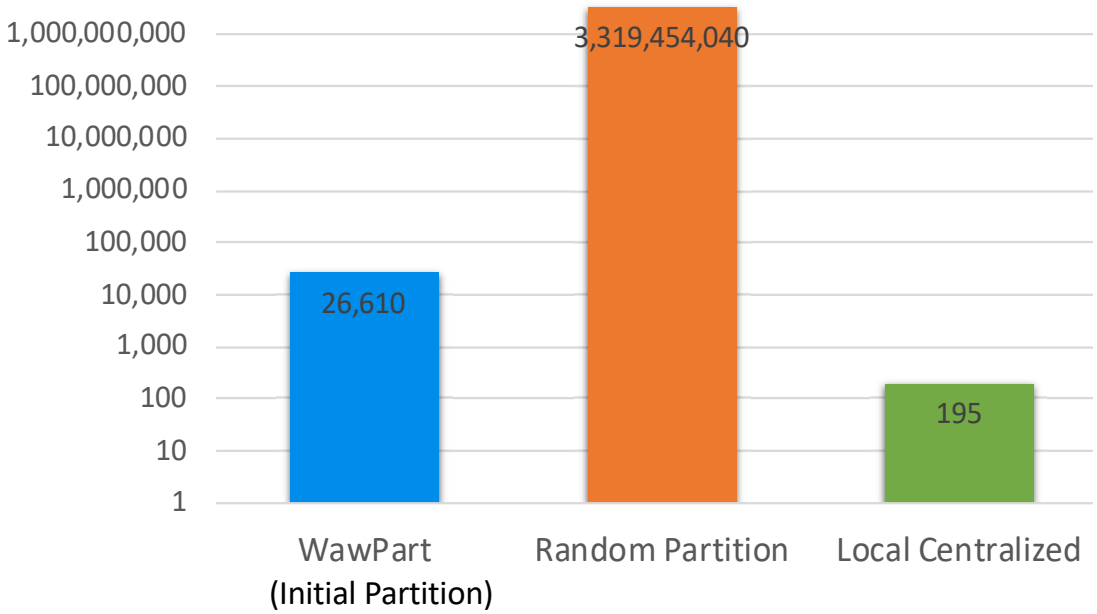
(**WawPart*** as Initial Partition)



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
WawPart	9	14	4	11	153	1206	11	207	159	4	12	7	5	960
Random Part	43	445	138	217	198	1260	72	8	469	180	4	33	210	975
millisecond	second	minutes	hour	days										

* A. Priyadarshi and K. J. Kochut, "WawPart: Workload-Aware Partitioning of Knowledge Graphs," Cham, 2021: Springer International Publishing, in Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices, pp. 383-395.

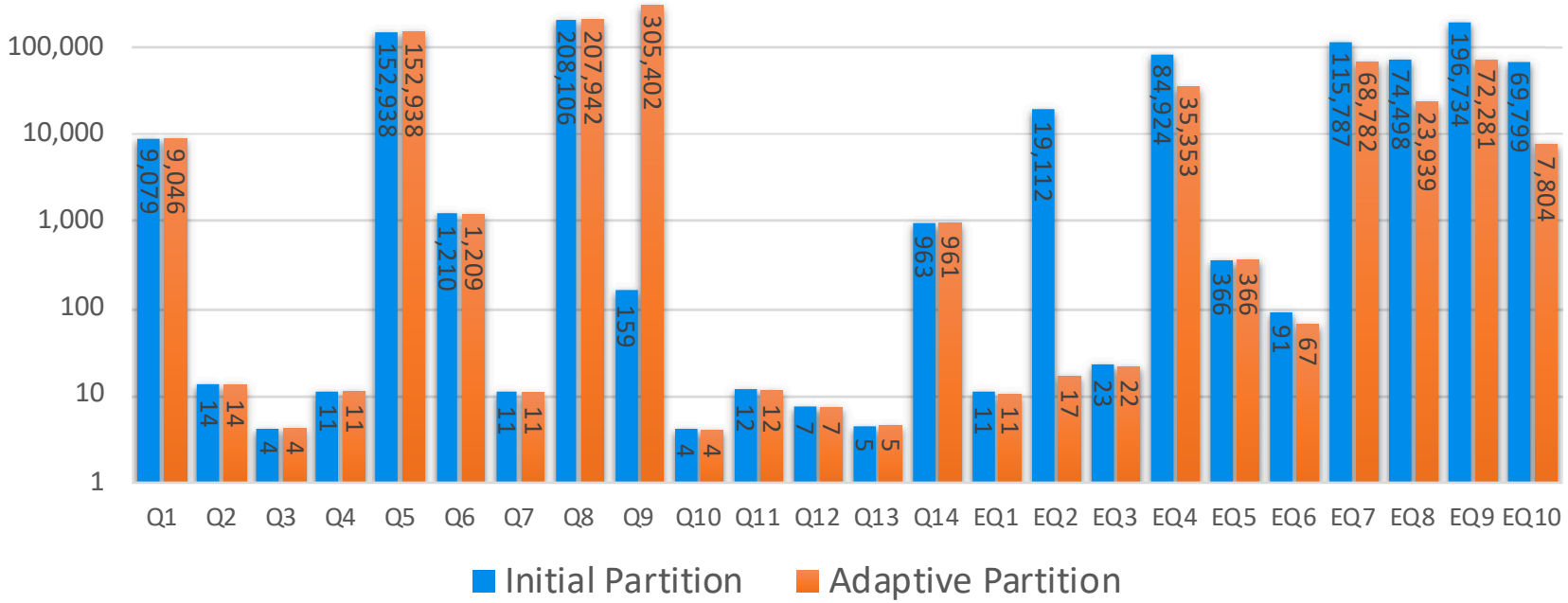
LUBM: Initial Partitioning



Average query execution time

- WawPart (Initial Partition): 26 second
- Random Part : 38 days
- Centralized: 195 - 198 milliseconds

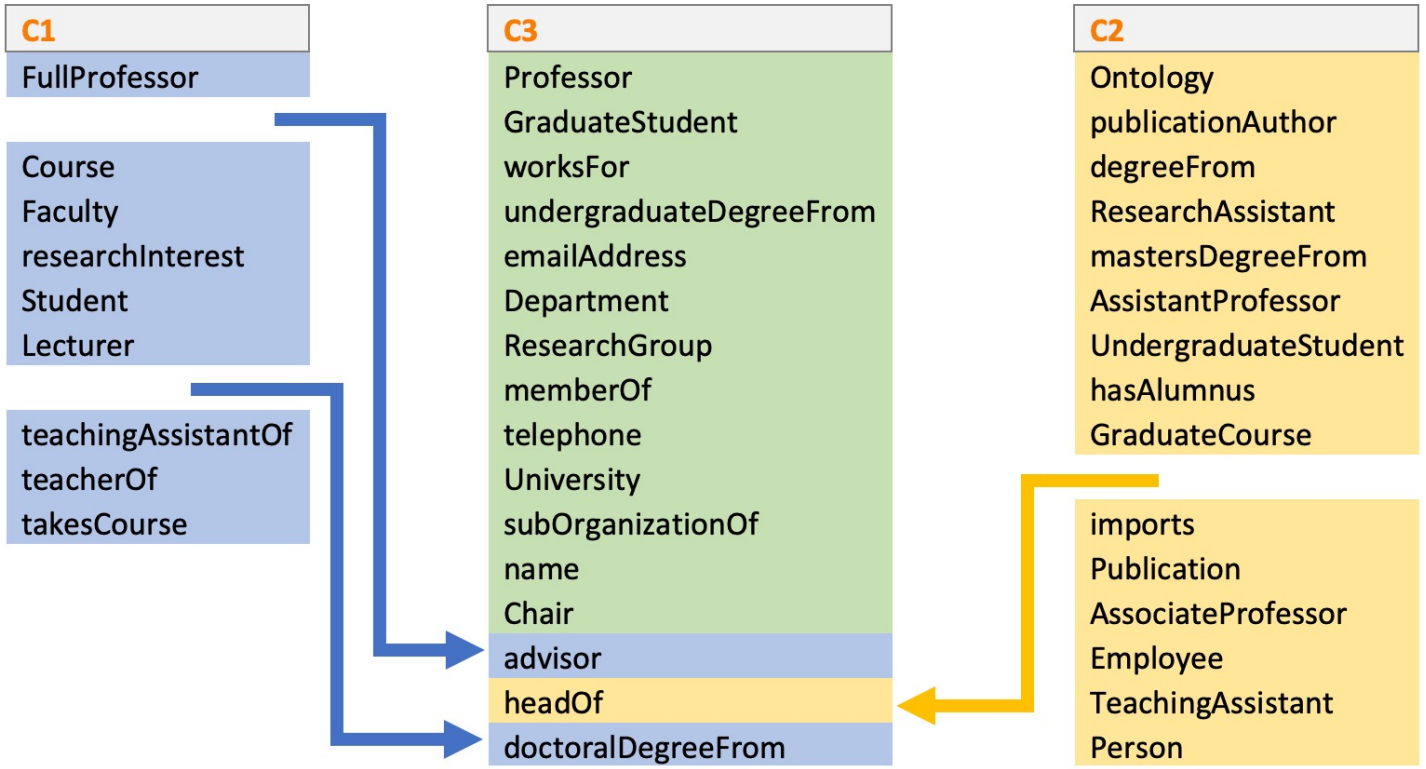
Exp 1: Change in composition of the workload



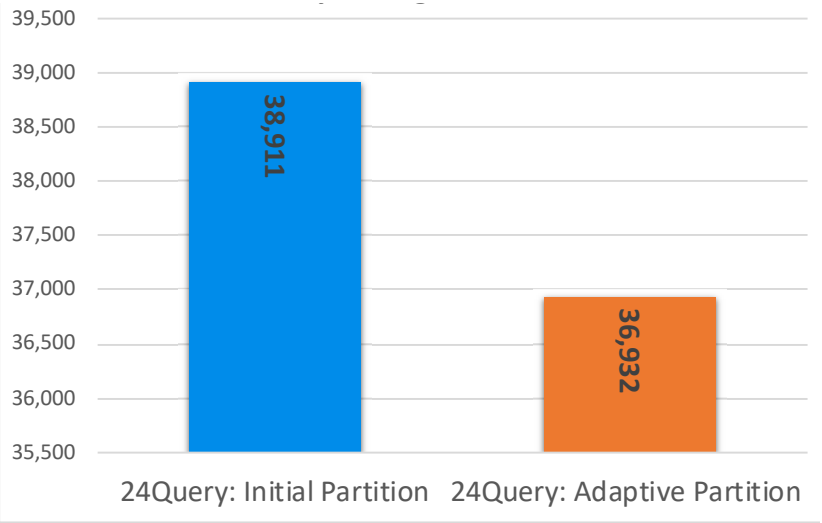
The modified workloads runtime on the initial partition and on our adaptive partitioning for the LUBM dataset were evaluated.

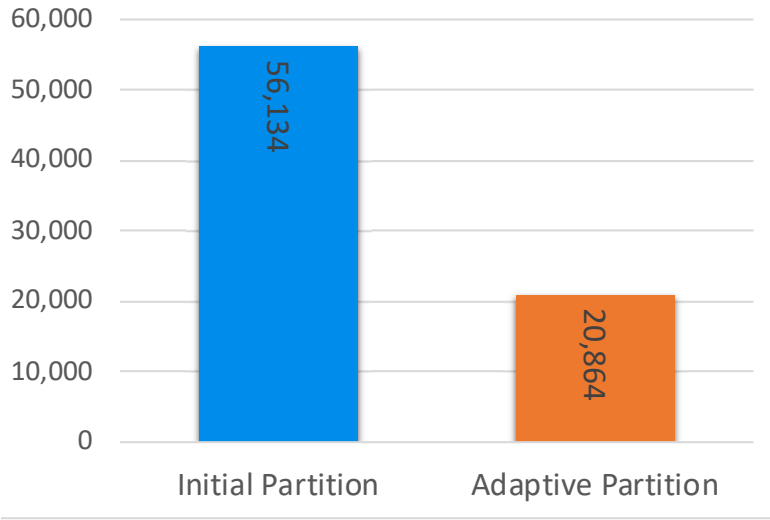
- EQ1 to EQ10 ten new queries
- Improved performance of new queries.

Movement of feature from one to another partition.



Exp1: 63% improvement in average runtime of the new workload





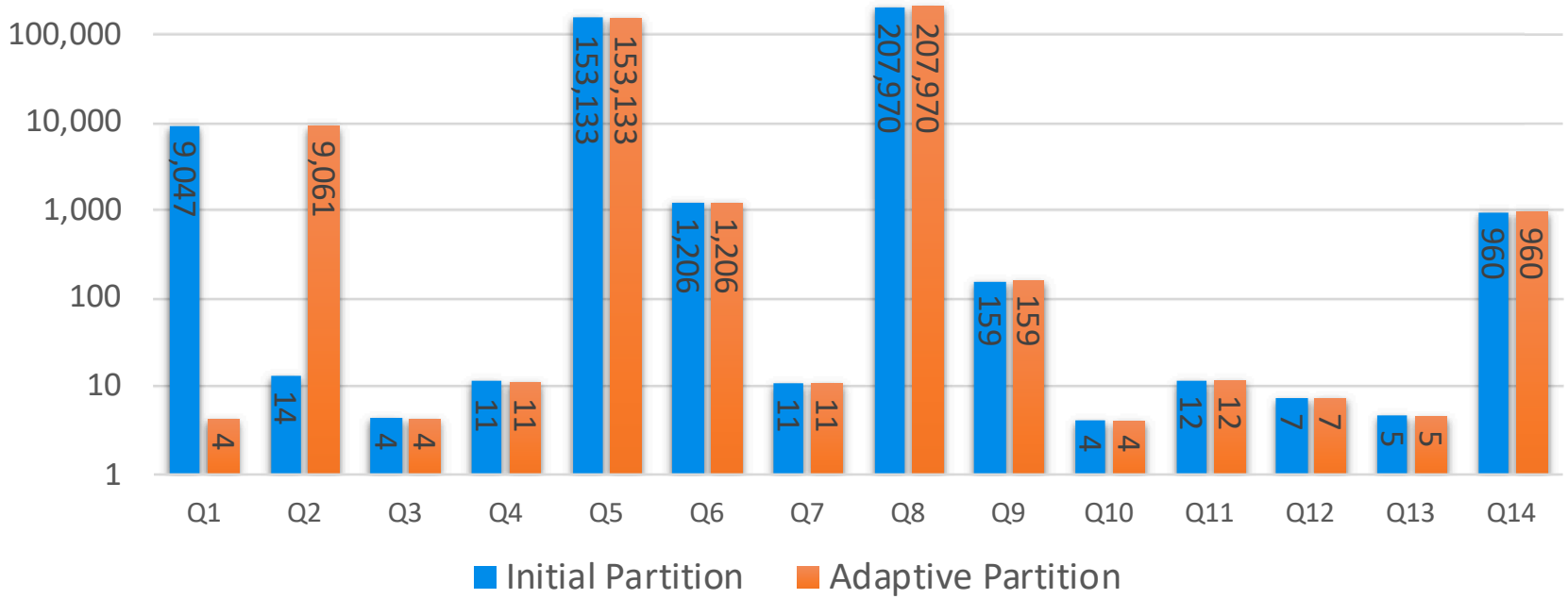
Average query execution time of 24 queries in millisecond

- Initial Partition : 38.9 second
- Adaptive Partition : 36.9 second

Average query execution time of 10 new queries in milliseconds

- Initial Partition : 56.1 second
- Adaptive Partition 20.8 second

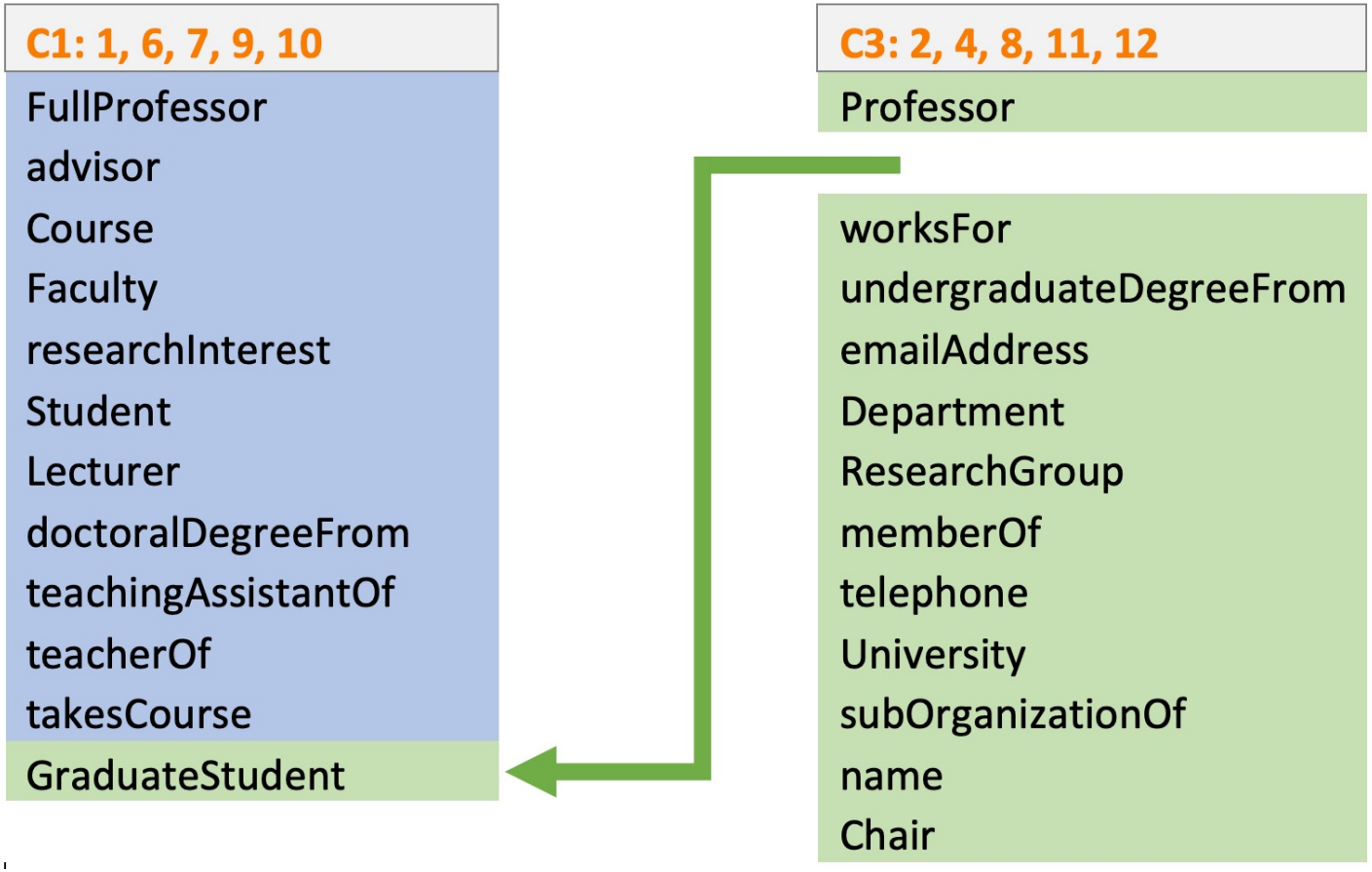
Exp2: Change in the relative frequency of queries in the workload



The changes in the relative frequency of queries in the workload executed on the initial partition as compared to the adaptive.

- if Q1 in LUBM is executed more frequently than the other 13 queries. The workload frequency share of query Q1 was increased to 50% of the whole workload.
- For Example, If all queries executed a total of 1000 times. Then Q1 executed 425 times and other 13 queries execute 45 times each approximately.

Movement of feature from one to another partition.



Exp2: Total workload execution time

Total workload execution time of queries in minutes

- Initial Partition : 329 minutes
- Adaptive Partition 272 minutes

Total workload execution time of queries in milliseconds

- Initial Partition : 19.7 million milliseconds
- Adaptive Partition 16.3 million milliseconds

Future Work

- To extend AWAPart into schema and Instance aware evolving knowledge graph adaptive re-partitioning system.
  - New entry of entity in knowledge graph
  - Old entry in knowledge graph is updated or deleted
  - Change in schema of Knowledge graph